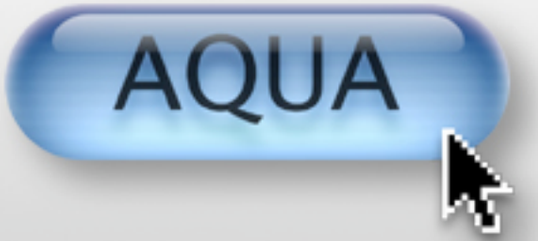
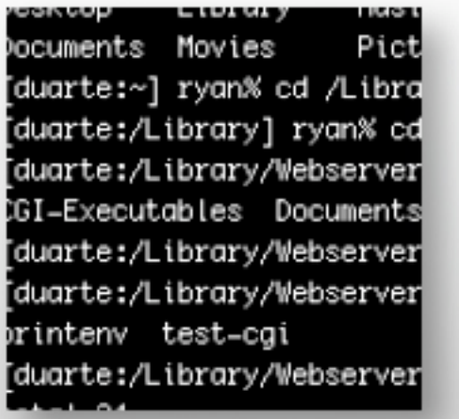
Introduction to Developing Applications With Cocoa

Session 400





Introduction to Developing Applications With Cocoa

Matthew Formica
DTS Engineer
mformica@apple.com

What You'll Learn

- What Cocoa is
- Cocoa benefits
- How to use it in all sorts of applications—including Carbon applications
- Cool Demos

What Is Cocoa

- Cocoa is a powerful, full-featured, easy-to-use, object-oriented framework for Mac OS X development
- Rapid development and increased productivity
- Fastest way to full-featured applications for Mac OS X

Mac OS X Architecture Diagram

Cocoa

Cocoa: Two Frameworks

Parts That Make a Whole

- Foundation and AppKit object-oriented Frameworks
- Powerful Development Tools
  - Project Builder IDE
  - Interface Builder
  - Performance Tools
- Powerful and dynamic languages: Objective-C/Java

Write Code Faster and Easier

- Framework APIs designed for extensibility, consistency
- Leverage Objective-C and the dynamic runtime

Write Code Faster and Easier

- Framework APIs designed for extensibility, consistency
- Leverage Objective-C and the dynamic runtime
  - Reuse more code
  - Good for creating shared frameworks—no ABI issues, can extend base classes
  - Categories—add methods to classes without subclassing

Be Faster to Market

- Spend your time developing your app's unique features
- Write less code—Cocoa keeps simple things simple, makes complex things possible
- Automatically get new features over time, without recompiling, reshipping

Which Language to Use?

- Cocoa APIs are in Objective-C (ObjC) and Java
- You can also plug your Carbon C and C++ code into ObjC
- Use the language that fits your needs
- We'll focus on ObjC today

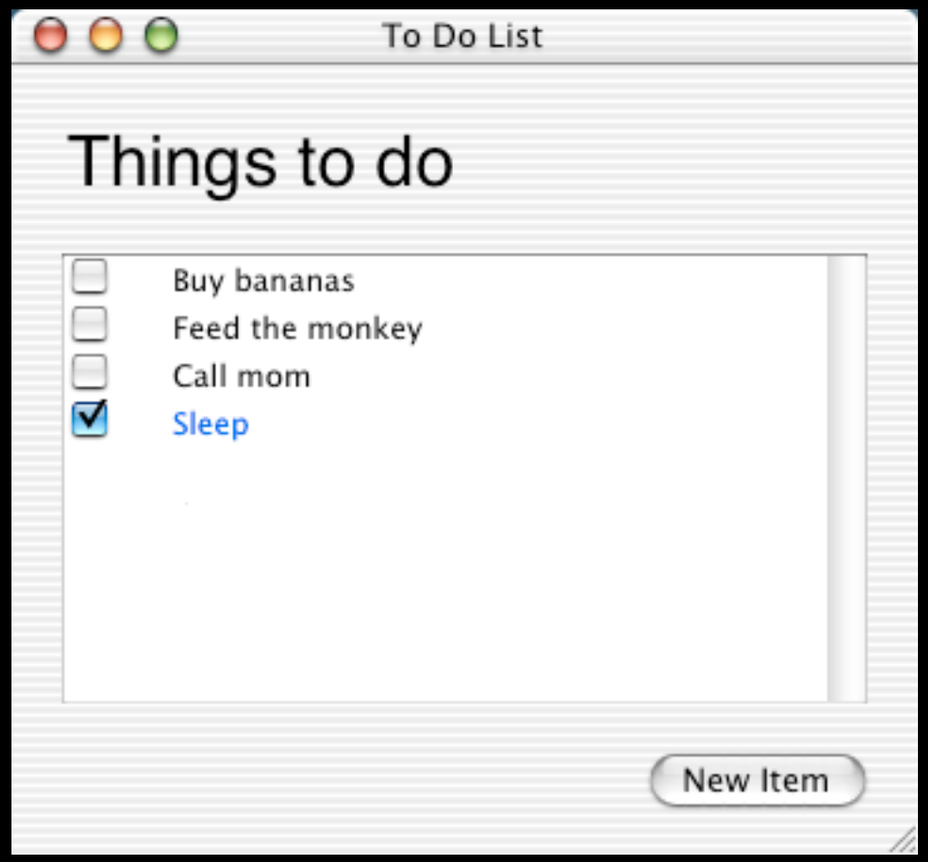
Cocoa & Objective-C Concepts

Tina Huang
Application Frameworks Engineer
tinahuang@apple.com

What You'll Learn

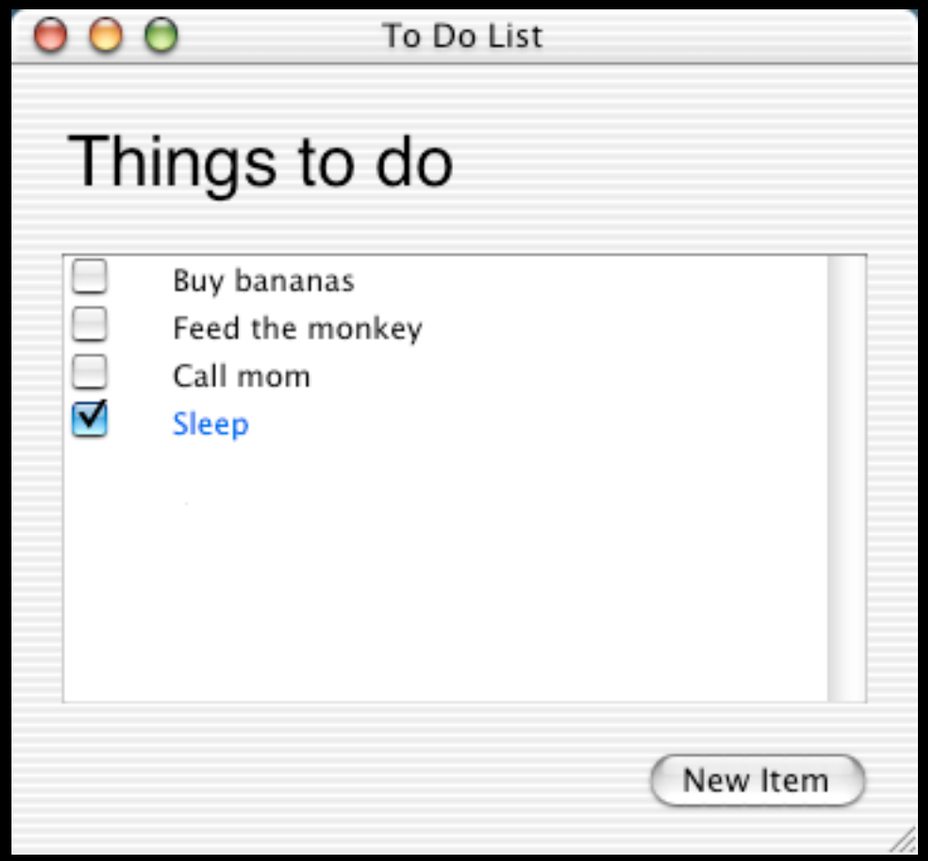
- Objective-C Syntax
- Cocoa Design Patterns
- Developer Tools

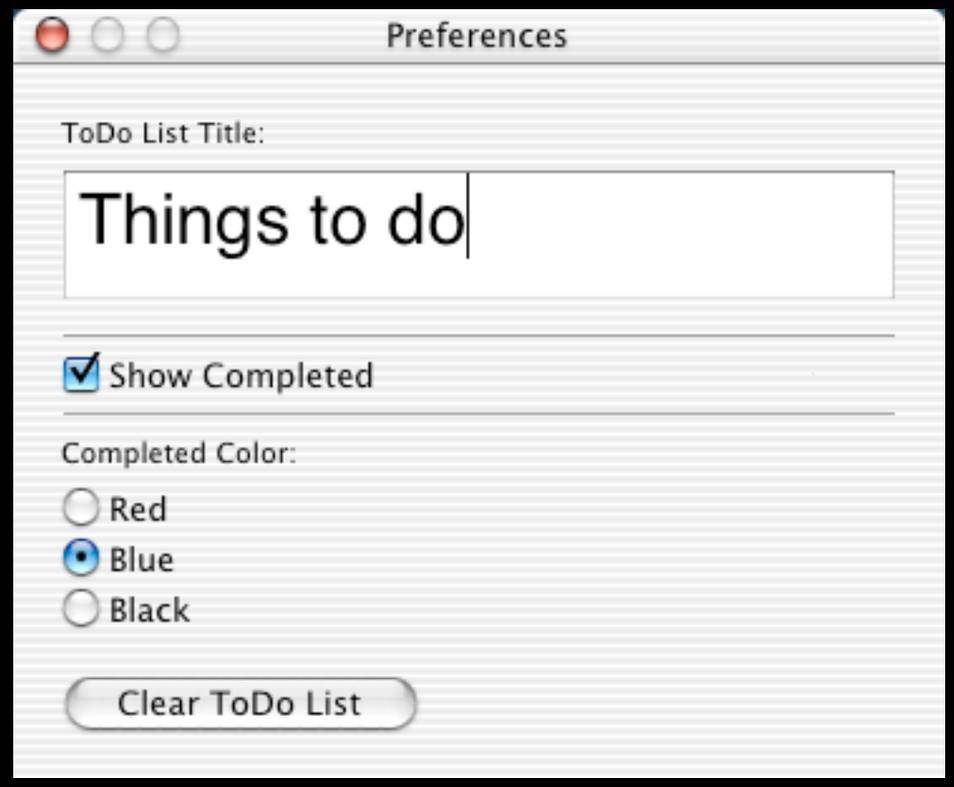
To Do List Example



To Do List Example

- Add a Preferences Pane





Objective-C Syntax

Valid Objective-C Code

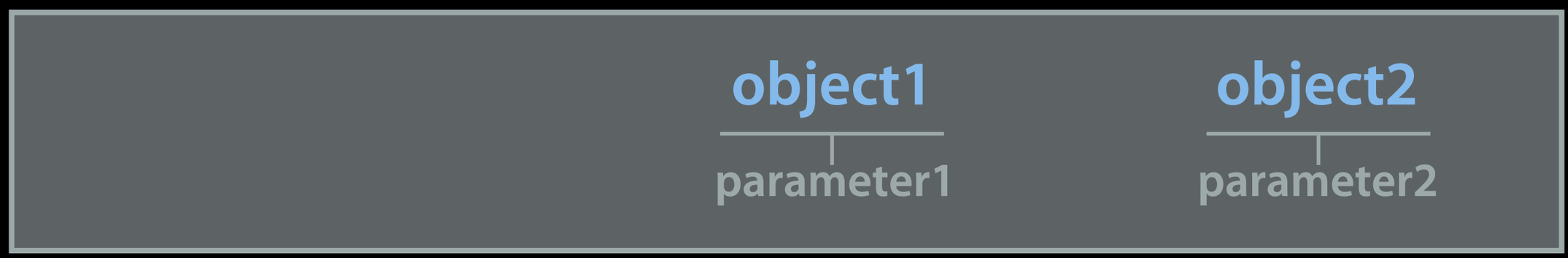
Objective-C Is a Superset of C

Additions to ANSI C

- Object Oriented Additions
- Messaging
  - Syntax for sending messages
  - Runtime support
  - Data types

Unique Messaging Syntax

```
myObject.doSomething ();
```




```
myObject.doWithThisAndThis (object1, object2);
```

```
id image = [[NSBitmapImageRep alloc] initWithBitmapDataPlanes: planes
                pixelsWide: 32 pixelsHigh: 32 bitsPerSample: 32
                samplesPerPixel: 4 hasAlpha: YES isPlanar: NO
                colorSpaceName: NSDeviceRGBColorSpace
                bytesPerRow: 0 bitsPerPixel: 0];
```

```
NSBitmapImageRep image = new NSBitmapImageRep(planes, 32, 32, 32, 4, true,
        false, DeviceRGBColorSpace, 0, 0);
```

Unique Messaging Syntax

```
[myObject doSomething];
```

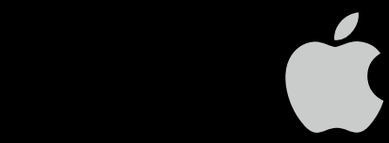
receiver — myObject

message — doSomething

```
[myObject  doWithThis:  object1  andThis:  object2 ];
```

parameter1 — object1

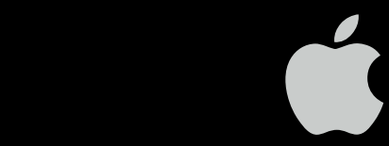
parameter2 — object2



Unique Messaging Syntax

```
doSomething
```

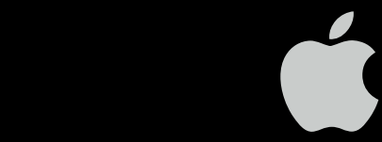
```
doWithThis:        andThis:
```



Unique Messaging Syntax

```
@selector (doSomething)
```

```
@selector (doWithThis: andThis:)
```



Dynamic Typing

- Object type determined at runtime
- Generic `id` object type
- Query an object to see its class and methods

Cocoa Design Patterns

Memory Management

- Cocoa uses reference counting
- `retain` increments the count, `release` decrements it, and `autorelease` decrements it later
- Objects are freed when the reference count drops to zero

The Memory Management Contract

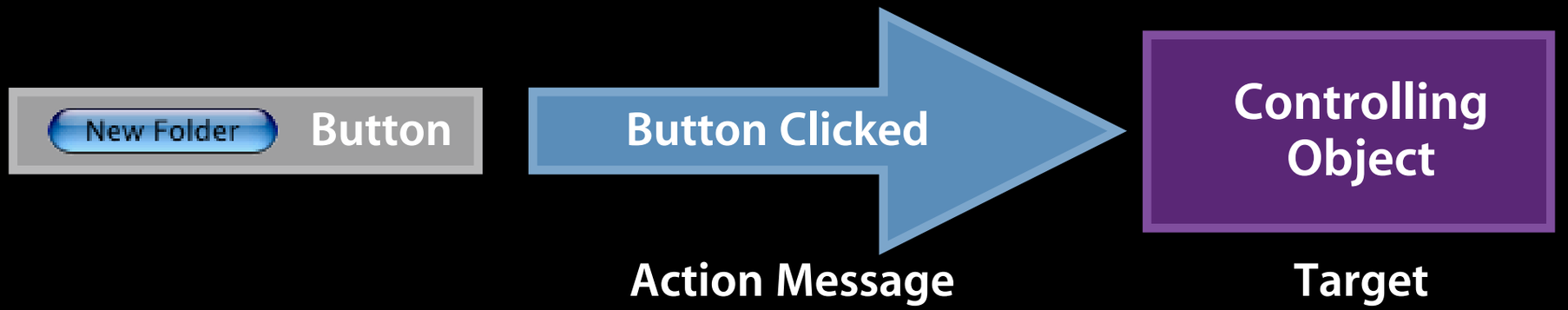
- Only creation methods return retained objects
- The receiver of an object is responsible for retaining and releasing
- The object is deallocated when there are no remaining references

Communicating With the UI

- Target/Action

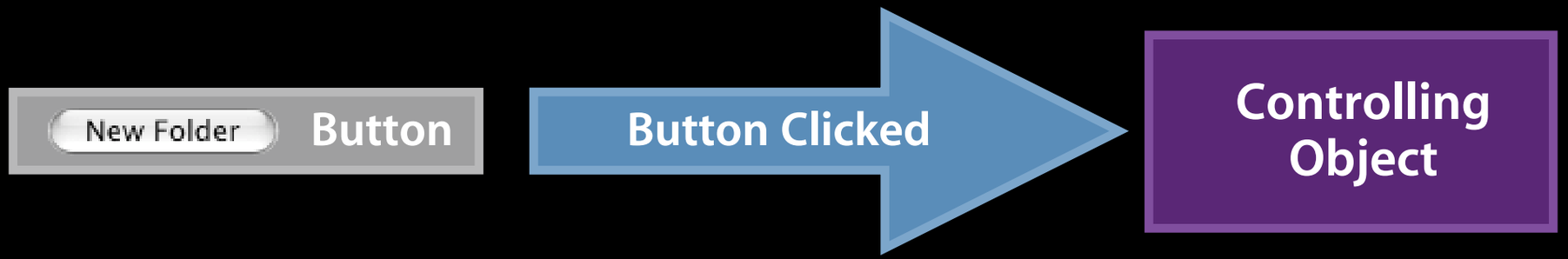
Communicating With the UI

- Target/Action

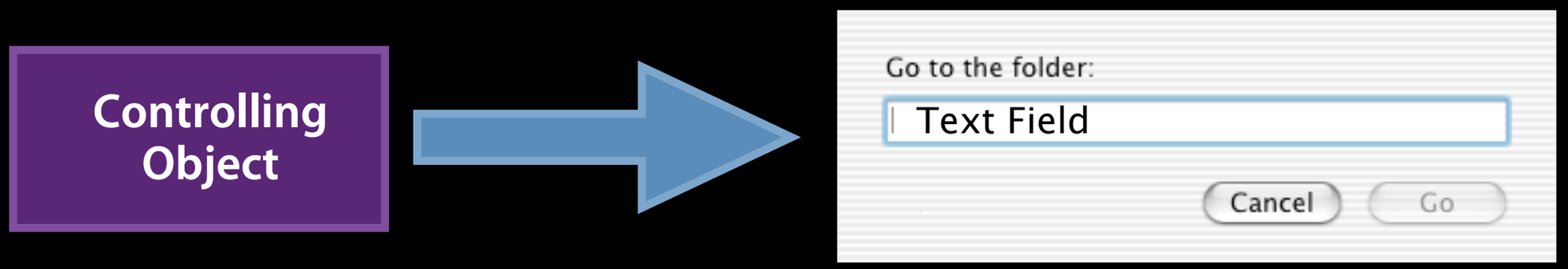


Communicating With the UI

- Target/Action



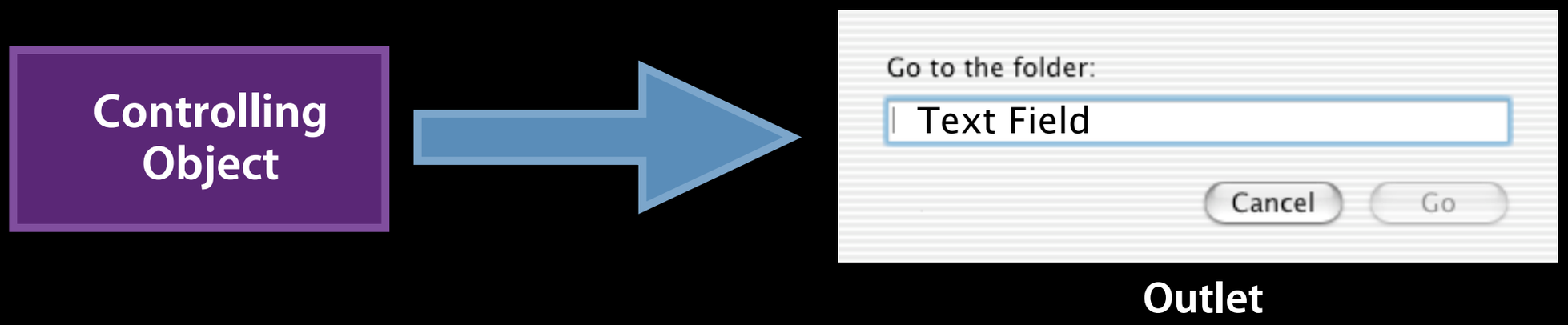
- Outlets



Communicating With the UI

- Target/Action

- Outlets



Model-View-Controller

- Model—stores and manipulates data
- View—displays information to the user
- Controller—connects the model to the view

Model-View-Controller

- Model—stores and manipulates data
- View—displays information to the user
- Controller—connects the model to the view

Interface Builder

- Quick and easy GUI Development
- Object, and object connection archiving
- Comprehensive Cocoa integration

Interface Builder Demo

Delegation

- An object can appoint another object to be its delegate
  - Modify the object's default behavior
  - Receives notifications of the object's events
- Several classes have a delegate outlet
- Customize object without subclassing

Delegation Demo

The Responder Chain

- Ordered list of objects that respond to events
- Target is determined on the fly
- First object in the responder chain that responds to a particular event is the First Responder

First Responder Examples

- Cut/Copy/Paste
- Menu item validation
- Font panel
- Color panel

First Responder Demo

Integrating Cocoa

Matthew Formica and Tina Huang

Mixing In Cocoa to Existing Apps

- Reasons for mixing Carbon and Cocoa
- How to mix C/C++ and/or Carbon with Cocoa
- Glypha III Demo: It's easy

Why Mix Carbon and Cocoa?

- Leverage existing code
- Write a common service
- Access alternative functionality

“Mix All the Ingredients in a Bowl…”

- Mix Carbon and Cocoa in the same source file
- Include Carbon.h or Cocoa.h, or both
- Change the file extension to .m or .mm as needed
- Use #ifdef __OBJC__

“Stir It All Up…”

- The OS provides "toll-free" bridging of common data types (e.g., CFString<->NSString)
- Mix non-GUI code prior to 10.2
- Mix GUI on a per-window basis in 10.2 and up
- Call NSApplicationLoad() to initialize Cocoa from a Carbon application

Glypha III

- Arcade game started in 1988
- Fully carbonized on SourceForge.net

Glypha III Demo

Glypha III

- Arcade game started in 1988
- Fully carbonized on SourceForge.net
- What we'll add:
  - Preference panel using Cocoa
  - Cocoa high score name entry dialog

Glypha III—Before the Integration

- Six source files
- Straight C
- Built-in basic sprite engine, sound engine
- High-score list, other preferences saved out manually

Handling Preferences

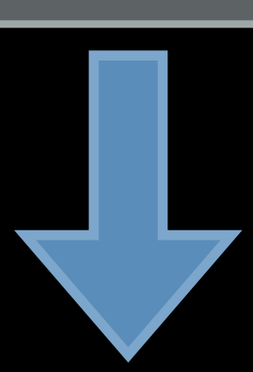
Direct Access

Using a Model

Carbon Modifications

- Pascal ANSII strings replaced with CFStrings
  - Enables easy data transfer to NSStrings
  - Adds Unicode support

```
PasStringCopy("\pYour Name", thePrefs.highName);
```



```
thePrefs.highName=CFSTR("Your Name");
```

Glypha III Demo

Summary

- Cocoa: faster development
- Cocoa fundamentals aren’t so hard
- Integrate Cocoa into existing applications

Wrap Up

Roadmap

410 Cocoa Update	Presidio Wed., 10:30am
412 Cocoa User Interface Programming	Presidio Wed., 3:30pm
413 Cocoa 2D Drawing Techniques	Nob Hill Wed., 5:00pm
419 Cocoa Performance Techniques	Russian Hill Fri., 9:00am

Roadmap

424 Cocoa: Tips and Tricks	Mission Fri., 2:00pm
427 Cocoa Text	Presidio Fri., 5:00pm
FF008 Cocoa Feedback Forum	North Beach Thur., 10:30am

Who to Contact

John Geleynse
Manager, Software Evangelism
geleynse@apple.com

Cocoa Feedback
For suggestions and feedback
cocoa-feedback@group.apple.com

Developer Technical Support
Code-level questions (fee-based)
dts@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

The Objective-C Programming Language
ADC Home > Documentation > Cocoa >

Program Design Programming Topic
ADC Home > Documentation > Cocoa >

Developing Cocoa Objective-C Applications: Tutorial
ADC Home > Documentation > Cocoa >

Reference Library

Documentation (Cont.)

Developing Cocoa Java Applications: A Tutorial
ADC Home > Documentation > Cocoa >

Integrating Carbon and Cocoa in Your Applications
ADC Home > Documentation > Cocoa >

Reference Library

Technical Notes

TN2005 “SimpleCocoaApp: An Overview”

Reference Library

Sample Code

Simple Cocoa Application
ADC Home > Sample Code > Cocoa > SimpleCocoaApp

Cocoa With Carbon or C++ Code
ADC Home > Sample Code > Cocoa > Cocoa_With_Carbon_Or_CPP

Cocoa GUI Code in a Carbon Application
ADC Home > Sample Code > Cocoa > CocoaInCarbon

Other Sample Code Projects
ADC Home > Sample Code > Cocoa

Reference Library

Other Resources

“Learning Cocoa”
Published by O’Reilly & Associates

“Cocoa In A Nutshell”
Published by O’Reilly & Associates

Other third-party introductions to Cocoa programming
Available in technical book stores

WWDC 2003

Worldwide Developers Conference